Lundi 20 janvier 2020

Lifelong Machine Learning et Deep Learning

Lifelong / Continuous / Incremental Learning

Présenté par:

Tony Gosse-Dumesnil
Jules Bourcier

M2 DAC REDS
Sorbonne Université
2019-2020

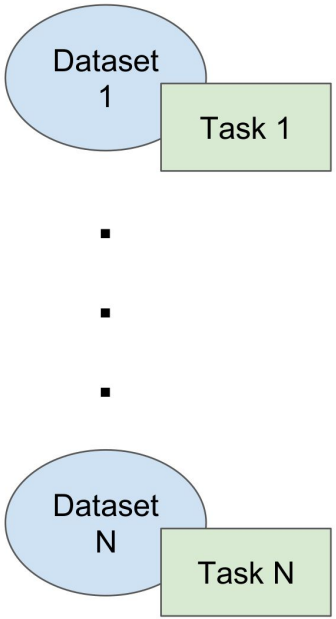
SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

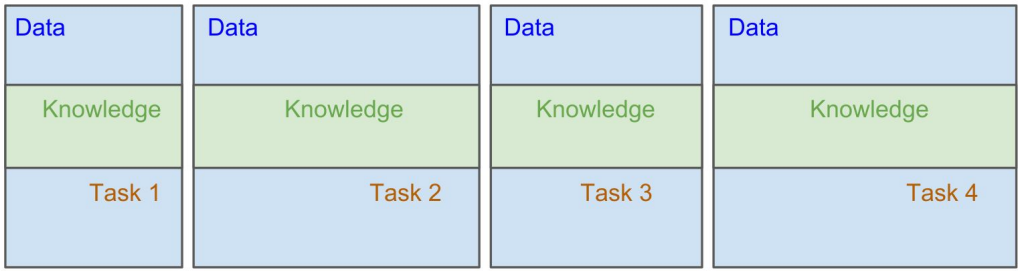
I. Introduction

Approche “classique” du Machine Learning

La plupart des modèles de ML ne peuvent qu’être entraîné dans dans un cadre restrictif

- Tâches simples et isolées
  - Tâches bien définies
  - Tâches apprises individuellement, séparément
  - Pas d’accumulation ni rétention de connaissances
- Toutes les données fournies avant l’apprentissage, dans un ordre quelconque
  - Beaucoup de données requises
- Mode batch
  - Pas de notion d’ordre temporel
- Hypothèse que les données et leur structure est statique





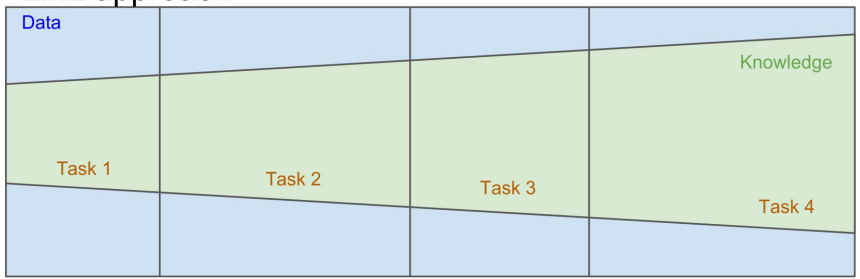
Lifelong ML / Continual Learning / Incremental Learning

Trois termes: même signification. Décrit en 1995

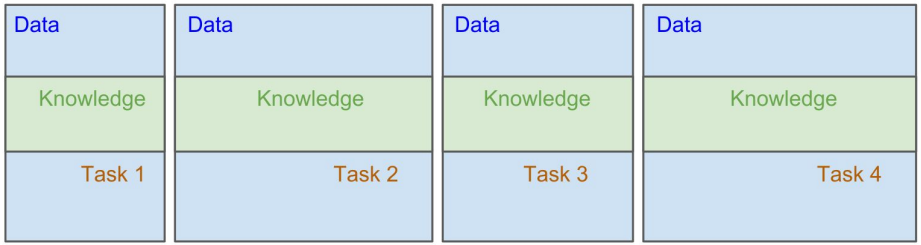
Définition: *Apprendre*, *conserver*, et *utiliser* de la connaissance, continuellement, sur une longue période de temps.

- Des flux de données arrivent constamment → apprentissage incrémental
- Multiple tâches à résoudre en séquence
- Conservation/accumulation de connaissances et utilisation des connaissances acc. pour aider de prochaines tâches
- Mime l'apprentissage humain

LML approach



'Classical' approach



S. Thrun. A lifelong learning perspective for mobile robot control, 1995.

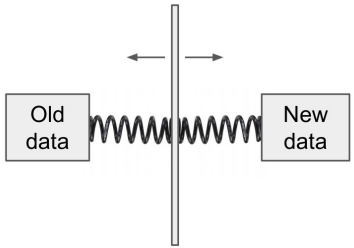
Challenges

Listes de challenges possibles auxquels il faut faire face:

- Les données ne sont pas disponible à priori, mais les exemples arrivent dans le temps
- Ressources mémoires limitées
  - Impossible de stocker et réentraîner sur toutes les données
  - Besoin de représentation compacte/implicite de la connaissance → réseaux de neurones bien adaptés
- Le modèle doit ajuster sa complexité
  - Impossible de déterminer la complexité / hyperparamètres en avance
  - Complexité bornée par les ressources disponibles
- *Concept drift*: changement dans le temps de la distrib. des données

- *Dilemme stabilité - plasticité:* quand et comment adapter le modèle courant ? Besoin de réactivité / adaptation & sans oublier l'ancienne information

  → Nécéssite d'un nouveau paradigme



Continual Learning (CL): Formalisation

On considère uniquement les réseau de neurones (NNs).

Soit un réseau de neurones $f(x,\Theta)$ paramétré par Θ. Données d'apprentissage d'un ensemble de tâches $\{ (X^{(1)}, Y^{(1)}), \ldots, (X^{(T)}, Y^{(T)}) \}$, entrées $X^{(t)} = \{ x^{(t,i)} \}_{i=1,\ldots nt}$, sorties $Y^{(t)} = \{ y^{(t,i)} \}_{i=1,\ldots nt}$ avec $nt = |X^{(t)}|$.

Tâches présentées séquentiellement: à l'instant t, seulement $(X^{(t)}, Y^{(t)})$ sont accessibles.

Lors de l'apprentissage de la tâche t, à partir des paramètres $\Theta^{(t-1)}$ et de $(X^{(t)}, Y^{(t)})$:

Le but est de trouver un nouveau set de paramètres $\Theta^{(t)}$ qui:

- (a) maintien la performance sur les tâches précédentes (< t) (éviter le *catastrophic forgetting*) comparé à $\Theta^{(t-1)}$
- ou bien: (b) l'améliore (*positive backward transfer*)
- et: (c) résoud la nouvelle tâche t en utilisant potentiellement la connaissance acquise précéd.^{mt} (*positive forward transfer*)

Parvenir à ces objectifs est non-trivial !

Trois propriétés d'un modèle de CL

Pour qualifier un algorithme de "continuous learner", trois propriétés sont requises :

i) Il doit être entrainable depuis un flux de données à partir duquel des exemples de différentes tâches se présentes à différents moments

ii) Il doit, à tout moment, être performant sur toutes les tâches observées

iii) Sa complexité en calcul et en mémoire doit être bornée, ou bien augmenter très lentement avec le nb. de tâches traitées

Prop. i) et ii): Essence du CL

Prop. iii): Applicabilité à un scénario du monde réel, avec potent.^{mt} un grand nb. de tâches en séquence.

Approches d'apprentissage connexes

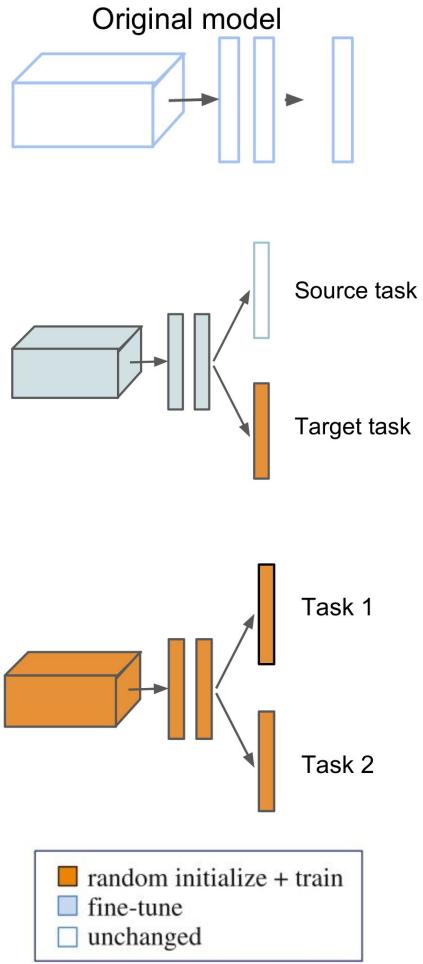
Approches similaires mais distinctes du CL :

- *Transfer learning (finetuning)*
  - Transfert unilatéral source → cible
  - La connaissance du domaine source aide à apprendre le domaine cible
  - Moins de données requises en cible
  - Les tâches doivent être similaires

  → Borne inférieure de perf. pour le CL

- *Multi-task learning*
  - Co-apprentissage simultané de tâches similaires
  - Chaque tâche traitée équitablement
  - But: optimiser la perf. sur toutes les tâches, via de la connaissance partagée
  - Plusieurs "têtes de réseau" (*multi-headed network*) et un extracteur commun

  → Borne supérieur de perf. pour le CL.



Catastrophic forgetting (CF)

Catastrophic forgetting - catastrophic interference:
Oubli, lors de l'apprentissage d'une nouvelle tâche, de connaissances clés requises pour résoudre une tâche précédente.
Dans les NNs: problème qui paraît inévitable et non résolu, identifié en 1989 [22]. Obstacle majeur au CL.

Pourquoi? Dans les NNs la connaissance est contenue dans les poids. La descente de gradient induit un déplacement des poids ce qui cause le CF.

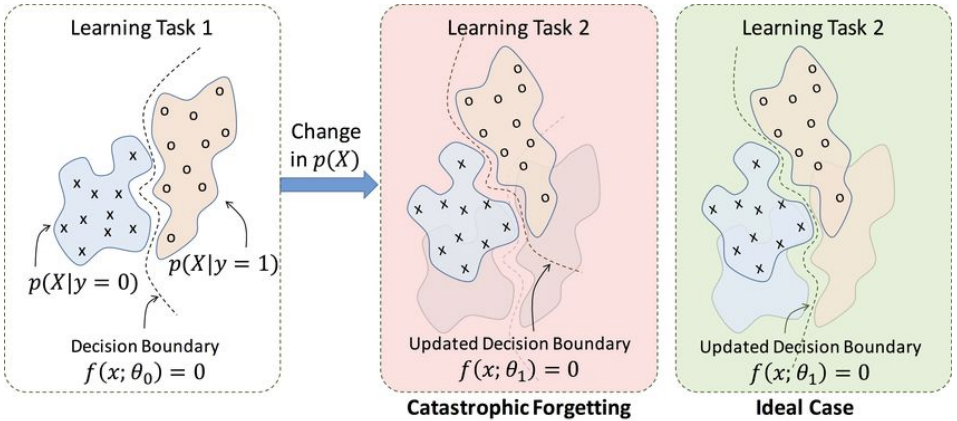
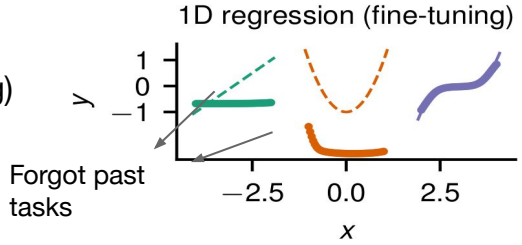


Figure 1: → L'apprentissage séquentiel (finetuning) cause du CF.



← Figure 2: Représentation schématique de CF causé par un changement de la distribution des données.

[22] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. 1989.

Comment le CF se mesure-t-il ?

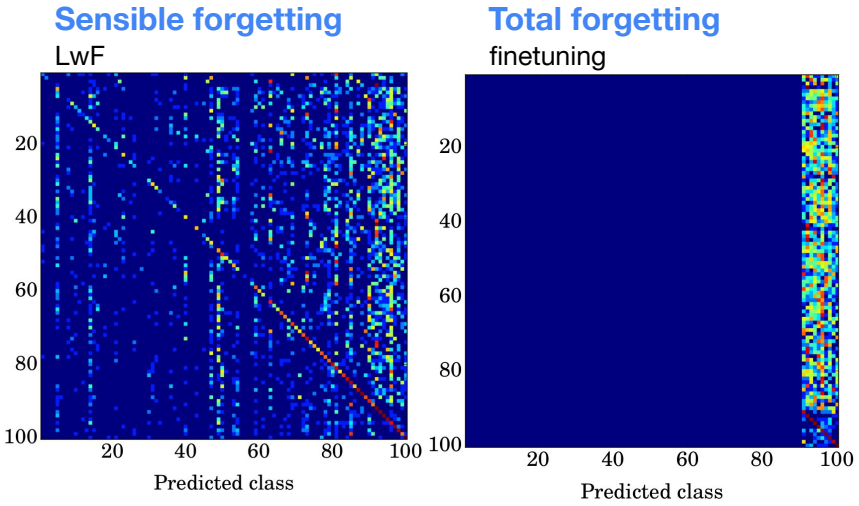


Figure 3: Matrices de confusion démontrant la présence de CF sur Incremental-CIFAR100.

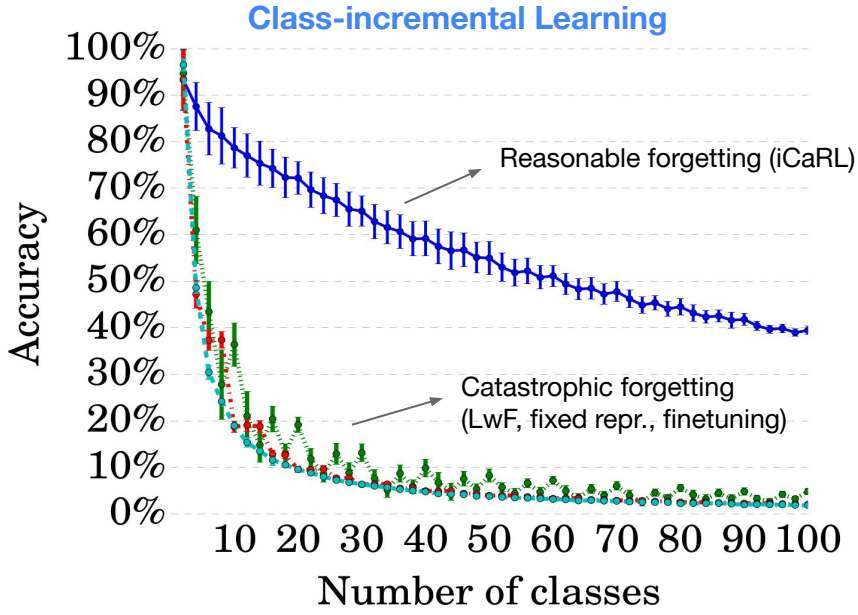


Figure 2: Accuracy multi-classe sur Incremental-CIFAR100 avec deux nouvelles classes par tâche. CF prononcé sur les courbes verte, rouge, bleu clair.

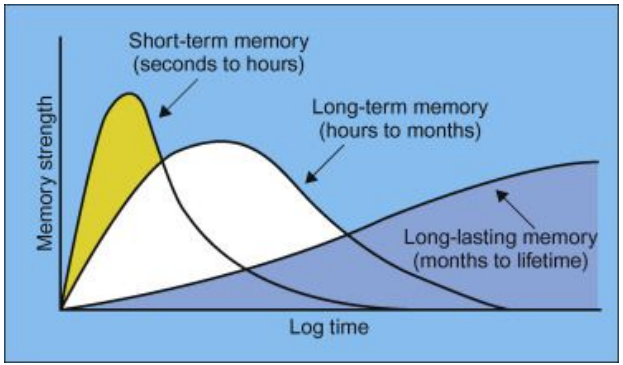
Analogie biologique: le cerveau

Contrairement aux NNs, les humains and d'autres animaux sont capable d'apprendre de façon continue.

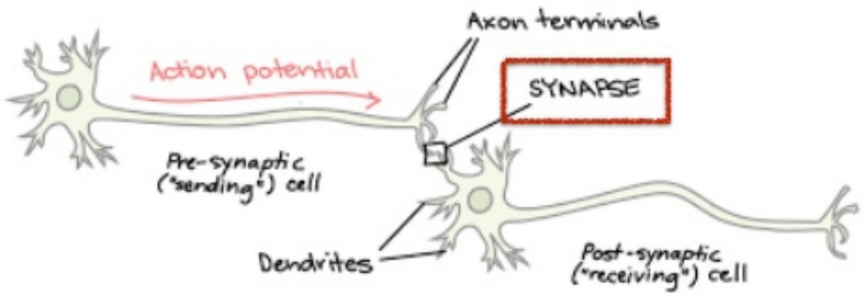
Le cerveau pourrait éviter le CF en protégeant le savoir acquis précédemment grâce à la *consolidation synaptique*: processus de renforcement de connexions neuronales

Dans le cerveau, un système de mémoire flexible est le résultat d'un bon équilibre entre *plasticité et stabilité synaptique*.

Types de mémoire humaine



Connexion entre neurones



Trois scénarios possible pour le Continual Learning

Trois scénarios par ordre de difficulté croissante.
Le plus simple est de connaître la tâche (scénario (1), Task-IL): possible d'utiliser des *multi-headed networks*.

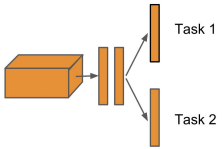


Table 1: Overview of the three continual learning scenarios.

Scenario	*Required at test time*
Task-IL	Solve tasks so far, task-ID provided
Domain-IL	Solve tasks so far, task-ID not provided
Class-IL	Solve tasks so far *and* infer task-ID

Dans le scénario (3, Class-IL): tâche associée à une classe, qu'il faut prédire.

[14] G. M. van de Ven and A. S. Tolias. Three scenarios for continual learning, april 2019.

II. État-de-l'art en Continuous Learning

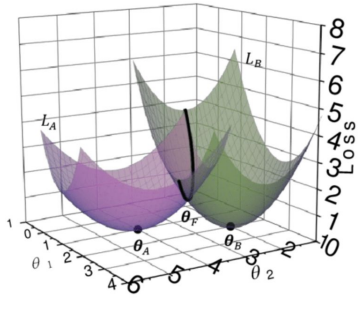
I. Optimisation régularisée / sous contraintes

Idea: Régulariser différemment chaque paramètre du réseau pendant l'app. de chaque nouvelle tâche

But: pénaliser les changements des poids estimé comme étant important pour les tâches précèd.^{mt} apprises, tout en donnant de la liberté aux poids moins importants.

Inspiré par la consolidation synaptique: stabilité renforcée de paramètres spécifiques à certaines tâches

Différentes méthodes pour calculer la contribution par paramètre à une tâche



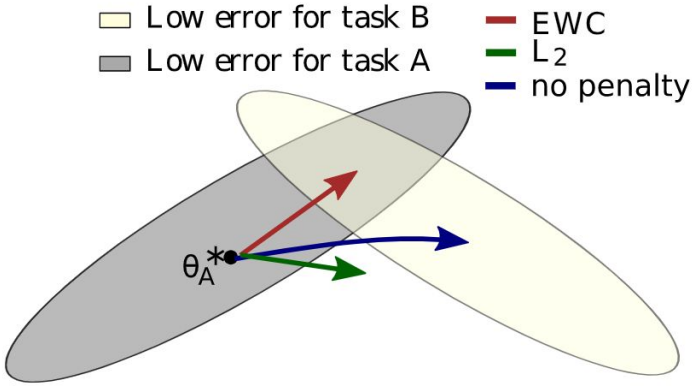


Figure: Trajectoire d'apprentissage illustrée dans un espace de paramètres schématique. Le but est de s'assurer que la tâche A n'est pas oubliée lors de l'entraînement sur la tâche B.

a. Elastic Weight Consolidation (EWC)

Importance des paramètres calculé via la matrice d'information de Fisher $F^{(k)}$ de coeff. diagonaux $\theta^{*(k)}$

Le terme de régularisation de EWC pour une tâche $K > 1$ est donné par:

$$\mathcal{L}^{(K)}_{\text{regularization}_{\text{EWC}}}(\boldsymbol{\theta}) = \sum_{k=1}^{K-1} \left(\frac{1}{2} \sum_{i=1}^{N_{\text{params}}} F_{ii}^{(k)} \left(\theta_i - \hat{\theta}_i^{(k)} \right)^2 \right)$$

Limites:

- Un terme quadratique pour chaque tâche: (k-1) termes
- Nécessite de stocker les $F_{ii}^{(k)}$ et $\theta^{*(k)}$ pour chaque tâche (k) (viole (iii))
- Calcul des $F^{(k)}$ coûteux, notamment si la tâche (k) possède bcp de données (possible compromis précision / coût en utilisant un sous-ensemble des données d'app.)

[2] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks, 2017.

b. Online EWC

Améliore EWC: Un seul terme de pénalisation quadratique ancré à $\theta^{*(K-1)}$ et importance des paramètres déterminée par une *decayed running sum* des matrices d'information de Fisher Information $F^{(k)}$ pour k <

Le terme de régularisation de online EWC pour une tâche K > 1 donné par:

$$\mathcal{L}^{(K)}_{\text{regularization}_{\text{oEWC}}} = \sum_{i=1}^{N_{\text{params}}} \tilde{F}_{ii}^{(K-1)} \left(\theta_i - \hat{\theta}_i^{(K-1)}\right)^2$$

En pratique, l'effet est similaire à celui d'EWC.

Limites:

- Calcul des $F^{(k)}$ coûteux, notamment si la tâche (k) possède bcp de données

[3] J. Schwarz, J. Luketina, W. M Czarnecki, A. Grabska-Barwinska, Y. Whye Teh, R. Pascanu, and R. Hadsell. Progress & compress: A scalable framework for continual learning, 2018.

c. Synaptic Intelligence (SI)

[4] F. Zenke, B. Poole, and S. Ganguli.
Continual learning through synaptic intelligence, 2017.

Similaire à online EWC, le terme de régularisation de SI consiste en un seul terme quadratique

Le terme de régularisation de SI pour une task K > 1 donné par:

$$\mathcal{L}^{(K)}_{\text{regularization}_{\text{SI}}} = \sum_{i=1}^{N_{\text{params}}} \Omega_i^{(K-1)} \left(\theta_i - \hat{\theta}_i^{(K-1)}\right)^2$$

En pratique, l'effet est similaire à celui d'online EWC.

Avantages et inconvénients de l'optimisation régularisée

Avantages:

- Peut être utilisé dans chacun des trois scénarios de CL;
- Facile à implémenter, à combiner à un autre type d'approche

Inconvénients:

- Échoue dans le scénario (3) Class-IL

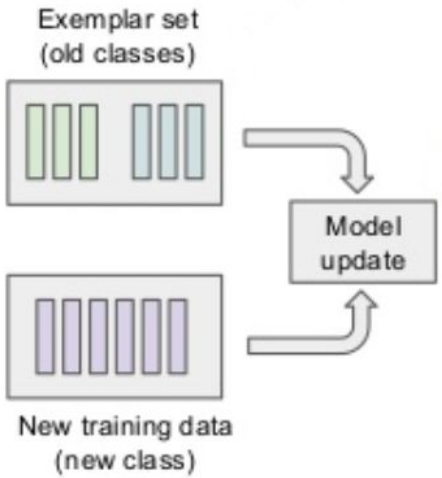
II. Replay / Rehearsal

Idée: Rejouer des données vu dans les tâches précédentes soit en stockant ces données de manière intelligente ou en les générant avec un modèle génératif

But: Préserver la réponse du réseau sur les tâches précédentes

Replay:
Essayes de reproduire les associations input-outputs des tâches précédentes à partir de “pseudo-data” des anciennes tâches (LwF, DGR)

Rehearsal:
Stocker et itérées des données précédentes de façon restreinte (iCaRL)

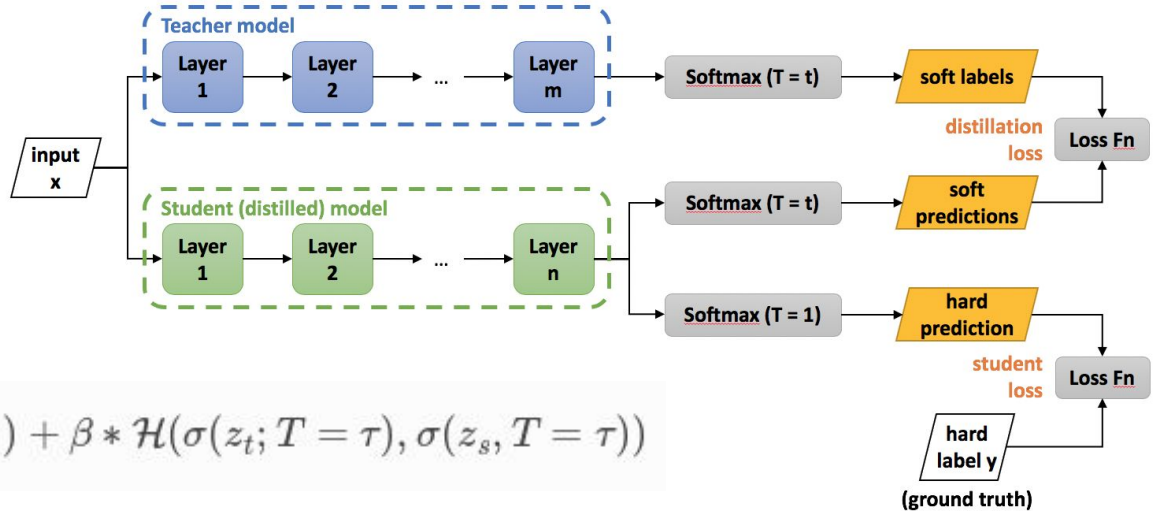


Distillation

Transférer de la connaissance d'un ensemble large de modèles et facile à apprendre dans un ensemble plus petit de modèles qui serait plus rapide et mieux adapté pour un déploiement.

- Dans la loss de distillation, on lisse la distribution des probabilité avec la température T dans le softmax pour augmenter l'influence des classes non prédites
- La loss standard et la loss de distillation sont combinés si on connait les vrais labels pour le transfert

$$p_i = \frac{exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)}$$

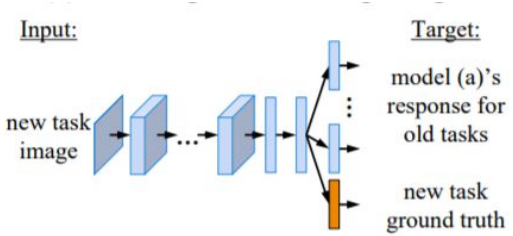


$$\mathcal{L}(x; W) = \alpha * \mathcal{H}(y, \sigma(z_s; T = 1)) + \beta * \mathcal{H}(\sigma(z_t; T = \tau), \sigma(z_s, T = \tau))$$

a. Learning without forgetting (LwF)

But: Ajouter de nouvelles tâches basées sur un partage adaptatif des paramètres sans accès aux données d'apprentissage des tâches précédentes.

Le learning without forgetting s'appuie sur les méthodes connues de multi-task learning, de feature extraction et de fine-tuning



LEARNING WITHOUT FORGETTING:

Start with:

θ_s: shared parameters

θ_o: task specific parameters for each old task

X_n, Y_n: training data and ground truth on the new task

Initialize:

$Y_o \leftarrow \text{CNN}(X_n, \theta_s, \theta_o)$ // *compute output of old tasks for new data*

$\theta_n \leftarrow \text{RANDINIT}(|\theta_n|)$ // *randomly initialize new parameters*

Train:

Define $\hat{Y}_o \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_o)$ // *old task output*

Define $\hat{Y}_n \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_n)$ // *new task output*

$$\theta_s^*,\ \theta_o^*,\ \theta_n^* \leftarrow \underset{\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n}{\text{argmin}} \left(\lambda_o \mathcal{L}_{old}(Y_o, \hat{Y}_o) + \mathcal{L}_{new}(Y_n, \hat{Y}_n) + \mathcal{R}(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n) \right)$$

[6] Z. Li and D. Hoiem. Learning without forgetting, 2017.

b. Incremental Classifier and Representation Learning (iCaRL)

iCaRL est une stratégie d'apprentissage qui permet d'apprendre de façon incrémentale des nouvelles classes qui arrivent progressivement dans le temps en évitant le catastrophic forgetting.

- Classification par technique de nearest-mean-of-exemplars
- Sélection des meilleurs exemples pour satisfaire la contrainte de mémoire
- Distillation de la connaissance des exemples des classes précédentes pour éviter le CF

$$y^* \leftarrow \underset{y=1,\dots,t}{\operatorname{argmin}} \|\varphi(x) - \mu_y\|$$

Algorithm 2 iCaRL INCREMENTALTRAIN

input $X^s, \dots, X^t$ // training examples in per-class sets
input K // memory size
require Θ // current model parameters
require $\mathcal{P} = (P_1, \dots, P_{s-1})$ // current exemplar sets
$\Theta \leftarrow$ UPDATEREPRESENTATION$(X^s, \dots, X^t; \mathcal{P}, \Theta)$
$m \leftarrow K/t$ // number of exemplars per class
for $y = 1, \dots, s-1$ **do**
$P_y \leftarrow$ REDUCEEXEMPLARSET(P_y, m)
end for
for $y = s, \dots, t$ **do**
$P_y \leftarrow$ CONSTRUCTEXEMPLARSET(X_y, m, Θ)
end for
$\mathcal{P} \leftarrow (P_1, \dots, P_t)$ // new exemplar sets

[8] S-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H Lampert. iCaRL: Incremental classifier and representation learning, 2017.

c. Deep Generative Replay (DGR)

Idée: entraîner un modèle génératif séparément, afin de générer les données à rejouer.

Techique: mixer données de la tâche courante avec données synthétiques des tâches passées
Modèle génératif: par ex. un VAE ou un GAN (conditionné à la tâche)

DGR protège le solveur du CF, et se protège aussi lui-même avec son propre replay

Hard targets: données gen. labelisées avec la catégorie la plus probable depuis une copie du solveur sur la tâche précédente.
Soft targets: données gen. appariées avec les sorties du softmax: DGR+distill (meilleur)

Avantages:

- Pas besoin de stocker des données passées
- Le modèle gen. se protège lui-même
- Bien que concept.^{mt} simple, souvent état de l'art sur les benchmarks de CL (split MNIST, permuted MNIST)

Inconvénients:

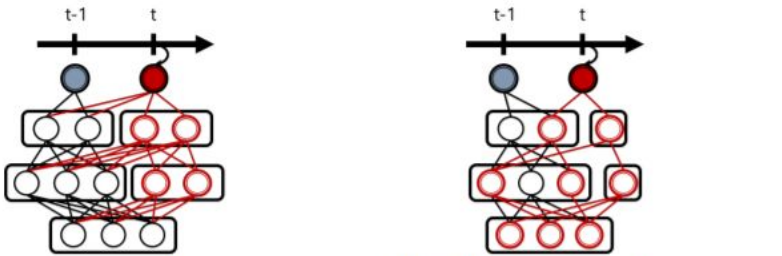
- Bon sur des datasets comme MNIST avec une distribution simple, mais pour des données complexes c'est bcp plus dur d'entraîner un générateur correct

III. Network Expansion

Idée: Augmentation de la capacité du réseau par l'ajout de neurones ou de couches.

Lorsque le réseau est saturé: solution est d'augmenter la capacité / expressivité.

/!\ N'inclut pas les multi-headed networks (scénario Task-IL)

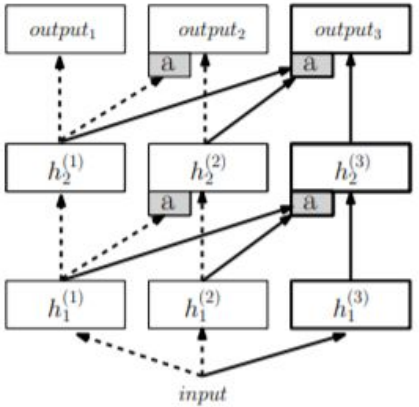


(b) No-retraining w/ expansion (c) Partial retraining w/ expansion

Progressive Networks (PNs)

Le progressive networks permet de faire du transfert de connaissance de tâche précédente en augmentant la taille du réseau à chaque nouvelle tâche.

- Un progressive network commence avec une seule colonne qui à L layers dont chaque layers possède des neurones.
- Quand une nouvelle tâche arrive, on ajoute une nouvelle colonne et chaque nouvelles layers recevront les entrées des layers de la couche précédente ainsi que les layers des tâches précédentes



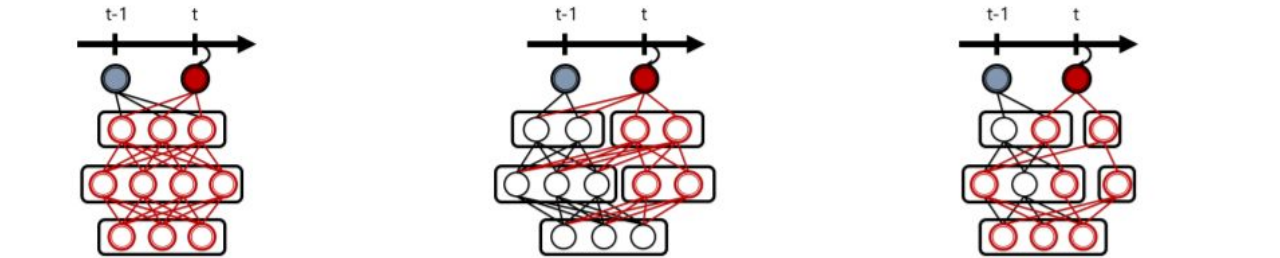
Limite:
- Le réseau grandit trop vite et est vite limité en mémoire (viole (iii))
- Augmentation constante,

$$h_i^{(k)} = f\left(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)}\right)$$

[16] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, Raia Hadsell. Progressive Neural Networks, 2016.

Dynamically Expandable Networks (DENs)

DEN est un réseau de neurones qui peut dynamiquement décider d'agrandir sa capacité en fonction d'un seuil et des nouvelles données des nouvelles tâches.

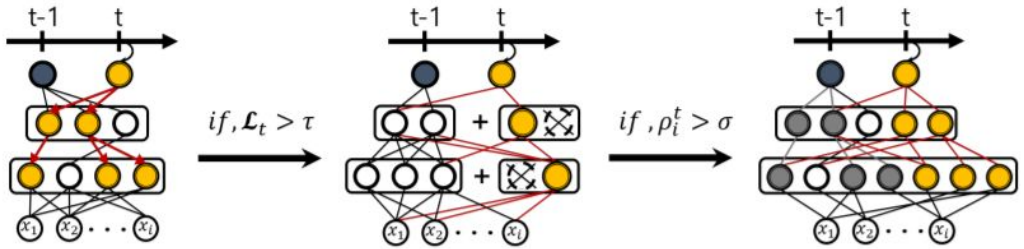


(a) Retraining w/o expansion (b) No-retraining w/ expansion (c) Partial retraining w/ expansion

- Gauche: réentrainement des poids du modèle (EWC)
- Milieu: Expansion du modèle et pas de réentrainement des poids (Progressive Network)
- Droite: Réentrainement partiel et expansion du réseau (DEN)

[15] Jaehong Yoon, Eunho Yang, Jeongtae Lee, Sung Ju Hwang. Lifelong Learning with Dynamically Expandable Networks, 2018.

Dynamically Expandable Networks (DENs)



$$\underset{\boldsymbol{W}^{t=1}}{\text{minimize}}\, \mathcal{L}(\boldsymbol{W}^{t=1}; \mathcal{D}_t) + \mu \sum_{l=1}^{L} \|\boldsymbol{W}_l^{t=1}\|_1$$

$$\rho_i^t = \|\boldsymbol{w}_i^t - \boldsymbol{w}_i^{t-1}\|_2$$

Procédé de l'algorithme:

- **Selective retraining:** DEN identifie les neurones qui sont important pour la nouvelle tâche et réentraine le réseau sur ces neurones.
- **Dynamic network expansion:** On agrandit le réseau en ajoutant k neurones sur chaque layer si la loss après le selective retraining est au dessus du seuil minimal. Puis on retire les neurones inutiles.
- **Network split/duplication:** On calcul le drift **ρi^t** pour chaque neurones pour identifier les neurones qui ont trop changé de leur valeur originale et on les duplique

[15] J. Yoon, E. Yang, J. Lee, S. Ju Hwang. Lifelong Learning with Dynamically Expandable Networks, 2018.

Task-conditioned Hypernetworks (TcH)

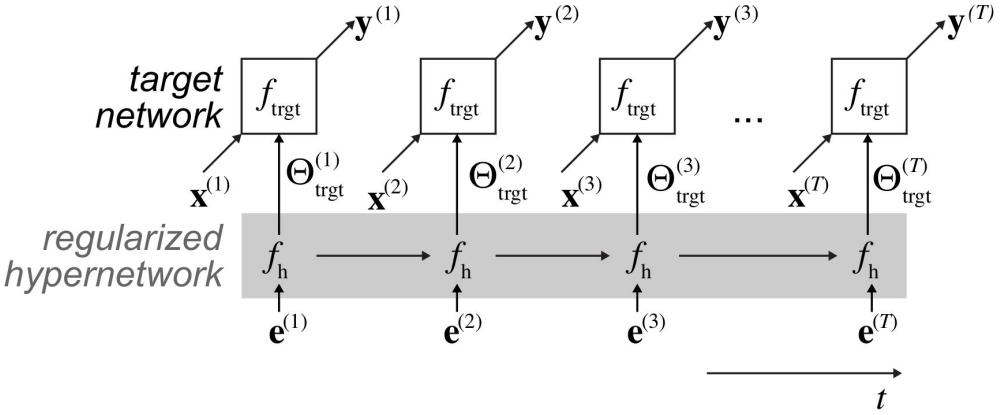
Idea: Instead of training directly a NN to solve sequence of tasks, train a hypernetwork.

Hypernetwork: A NN that takes a context info. as input and generates weights for another NN (the one that solves the task).

Task-conditional meta-modeling:
Utiliser un "metamodèle": étant donné un embedding d'une tâche $e^{(T)}$, génère les poids $\Theta^{(T)}_{\text{target}}$ du réseau cible (le solveur)

Les embeddings de tâches $e^{(T)}$ donne un contexte dépendant et spécifique à une tâche

Approche singulière et innovante

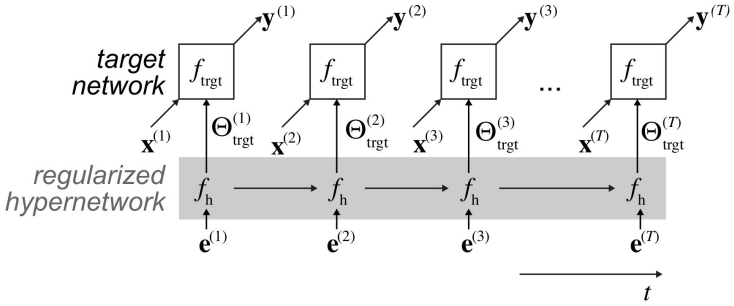


[11] D. Ha, A. M. Dai, and Q. V. Le. Hypernetworks, 2017.
[12] J. von Oswald, C. Henning, J. Sacramento, B. F. Grewe Continual learning with hypernetworks, october 2019.

Task-conditioned Hypernetworks (TcH)

But: apprendre $\Theta_h^{*(T)}$ et $\{ e^{(t)} \}_{t = 1..T}$

Régularisateur sur la sortie: pénalise les changements sur $\Theta^{(t)}_{target}$ pour t < T (t termes quadratiques sur les poids $\Theta^{(t)}_{target}$)



Optimisation en deux étapes:

Step 1: calculer $\Delta\Theta_h$ qui minimise la loss de la tâche courante:

$\Delta\Theta_h$ = argmin $\mathcal{L}_{\text{task}}^{(T)} = \mathcal{L}_{\text{task}}(\Theta_{\text{h}}, \mathbf{e}^{(T)}, \mathbf{X}^{(T)}, \mathbf{Y}^{(T)})$

Step 2: minimiser la loss totale (loss tâche + contrainte sorties)

$\Theta_h^{*(T)}$ = argmin $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}}(\Theta_{\text{h}}, \mathbf{e}^{(T)}, \mathbf{X}^{(T)}, \mathbf{Y}^{(T)}) + \mathcal{L}_{\text{output}}(\Theta_{\text{h}}^{*}, \Theta_{\text{h}}, \Delta\Theta_{\text{h}}, \{\mathbf{e}^{(t)}\})$

$$= \mathcal{L}_{\text{task}}(\Theta_{\text{h}}, \mathbf{e}^{(T)}, \mathbf{X}^{(T)}, \mathbf{Y}^{(T)}) + \frac{\beta_{\text{output}}}{T-1} \sum_{t=1}^{T-1} \| f_{\text{h}}(\mathbf{e}^{(t)}, \Theta_{\text{h}}^{*}) - f_{\text{h}}(\mathbf{e}^{(t)}, \Theta_{\text{h}} + \Delta\Theta_{\text{h}}) \|^2$$

$e^{(T)}$: Task-specific Embedding de la tâche est appris par backprop conjoint^{mt} à $\Theta_h^{*(T)}$.

IV. Évaluation en Continual Learning

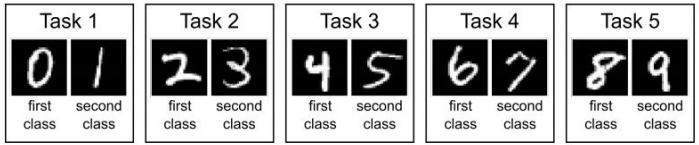


Figure 1: Schematic of split MNIST task protocol.

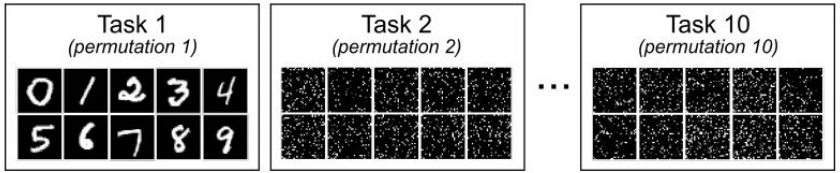


Figure 2: Schematic of permuted MNIST task protocol.

Table 2: Split MNIST according to each scenario.

Task-IL	With task given, is it the 1st or 2nd class? (e.g., 0 or 1)
Domain-IL	With task unknown, is it a 1st or 2nd class? (e.g., in $[0, 2, 4, 6, 8]$ or in $[1, 3, 5, 7, 9]$)
Class-IL	With task unknown, which digit is it? (i.e., choice from 0 to 9)

Table 3: Permuted MNIST according to each scenario.

Task-IL	Given permutation X, which digit?
Domain-IL	With permutation unknown, which digit?
Class-IL	Which digit *and* which permutation?

[14] G. M. van de Ven and A. S. Tolias. Three scenarios for continual learning, april 2019.

Résultats des méthodes sur les trois scénarios décrits

Table 1: Accuracy moyenne en test (sur toutes les tâches) sur le protocole *split MNIST*.

Table 2: Idem sur le protocole *permuted MNIST*

[14] G. M. van de Ven and A. S. Tolias. Three scenarios for continual learning, april 2019.

Table 4: Average test accuracy (over all tasks) on the split MNIST task protocol. Each experiment was performed 20 times with different random seeds, reported is the mean (± SEM) over these runs.

Approach	Method	Task-IL	Domain-IL	Class-IL
Baselines	*None – lower bound*	*87.19 (± 0.94)*	*59.21 (± 2.04)*	*19.90 (± 0.02)*
	Offline – upper bound	*99.66 (± 0.02)*	*98.42 (± 0.06)*	*97.94 (± 0.03)*
Task-specific	XdG	99.10 (± 0.08)	-	-
Regularization	EWC	98.64 (± 0.22)	63.95 (± 1.90)	20.01 (± 0.06)
	Online EWC	99.12 (± 0.11)	64.32 (± 1.90)	19.96 (± 0.07)
	SI	99.09 (± 0.15)	65.36 (± 1.57)	19.99 (± 0.06)
Replay	LwF	99.57 (± 0.02)	71.50 (± 1.63)	23.85 (± 0.44)
	DGR	99.50 (± 0.03)	95.72 (± 0.25)	90.79 (± 0.41)
	DGR+distill	99.61 (± 0.02)	96.83 (± 0.20)	91.79 (± 0.32)
Replay + Exemplars	iCaRL (budget = 2000)	-	-	94.57 (± 0.11)

Table 5: Idem as Table 4, except on the permuted MNIST task protocol.

Approach	Method	Task-IL	Domain-IL	Class-IL
Baselines	*None – lower bound*	*81.79 (± 0.48)*	*78.51 (± 0.24)*	*17.26 (± 0.19)*
	Offline – upper bound	97.68 (± 0.01)	97.59 (± 0.01)	97.59 (± 0.02)
Task-specific	XdG	91.40 (± 0.23)	-	-
Regularization	EWC	94.74 (± 0.05)	94.31 (± 0.11)	25.04 (± 0.50)
	Online EWC	95.96 (± 0.06)	94.42 (± 0.13)	33.88 (± 0.49)
	SI	94.75 (± 0.14)	95.33 (± 0.11)	29.31 (± 0.62)
Replay	LwF	69.84 (± 0.46)	72.64 (± 0.52)	22.64 (± 0.23)
	DGR	92.52 (± 0.08)	95.09 (± 0.04)	92.19 (± 0.09)
	DGR+distill	97.51 (± 0.01)	97.35 (± 0.02)	96.38 (± 0.03)
Replay + Exemplars	iCaRL (budget = 2000)	-	-	94.85 (± 0.03)

III. Notre idée de contribution

Motivation et intuition

Idée: Reprendre et améliorer la stratégie vue dans les task-conditioned hypernetworks (TcH),
avec pour objectif d'inférer une tâche lorsque inconnue, ceci de façon efficace (et complex. calcul borné non dépendante du nb. de tâches)

Dans TcH: utilisation d'un "task-inference classifier", prédisant l'id. de la tâche, et régularisé (type EWC)

énérer les paramètres des tâches et l'a combiner avec du DGR pour diminuer le CF.

Intuition:

- Au lieu d'utiliser un task-inference-classifier, apprendre un "task-embeddings regressor" (f_{TI}) qui infère directement un l'embedding e(x) à partir d'un input x.
- Appliquer DGR pour protéger f_{TI} par synthetic replay, et DGR et protégé par un hypernetwork (similairement au solveur protégé dans TcH)

Titre d'article: *Continuous Learning with Task-embeddings inference supported by hypernetwork-protected synthetic replay*

Description

Précédemment (TcH): “task-inference classifier”

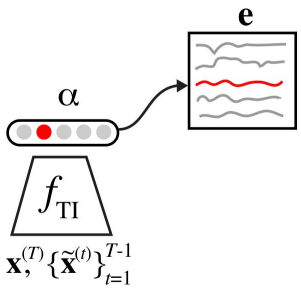
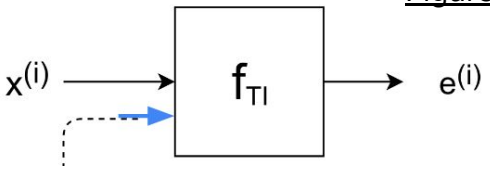


Figure 1: Réseau d’inférence de tâche f_{TI}

Maintenant: “task-embeddings regressor”:



Protection par *deep-generative replay* (DGR).
Hypernetwork-protected generator

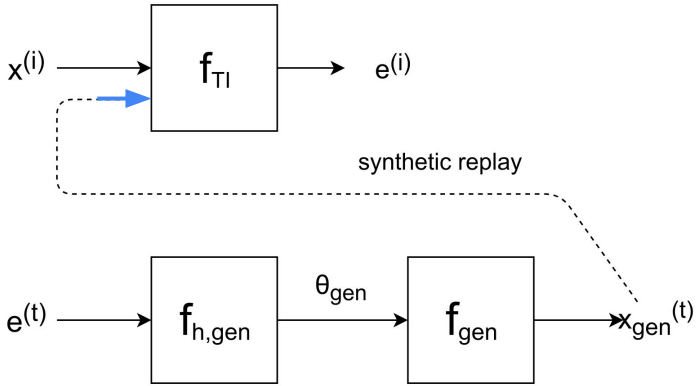


Figure 2: Réseau d’inférence de tâche f_{TI} et “hypernetwork-protected replay generator” f_{gen}

Description

Le réseau target (solveur) est appris comme dans TcH:

Figure 2: Hypernetwork f_h pour le réseau solveur de tâche f_{target}

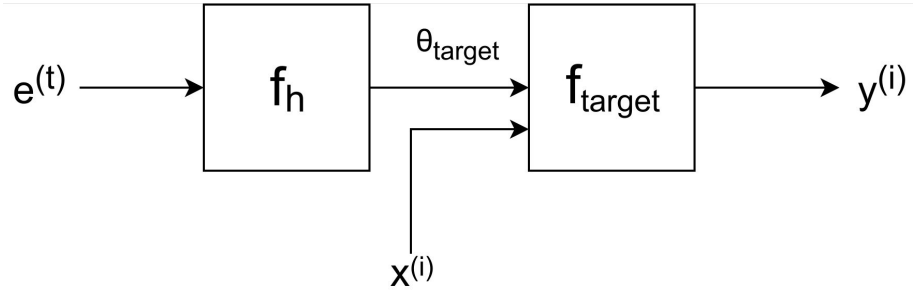
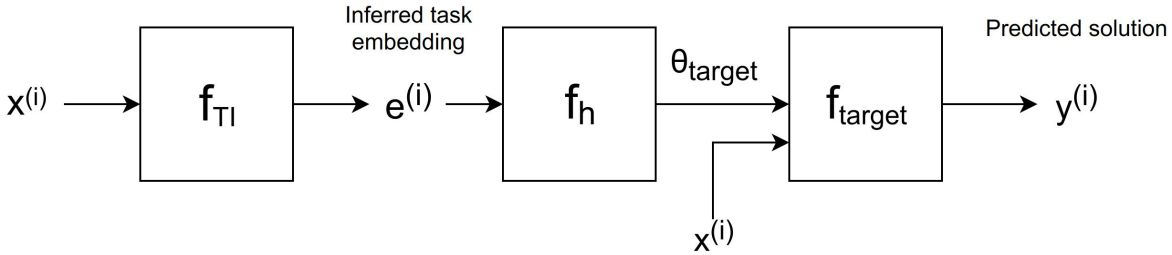


Figure 3: Réseau d'inférence de tâche f_{target}

Inférence avec notre modèle: pipeline fluide et efficace

Figure 3: Inférence avec notre modèle



Protocole d'évaluation

Table 1: Accuracy moyenne en test (sur toutes les tâches) sur le protocole *split MNIST*.

Table 4: Average test accuracy (over all tasks) on the split MNIST task protocol. Each experiment was performed 20 times with different random seeds, reported is the mean (± SEM) over these runs.

Approach	**Method**	**Task-IL**	**Domain-IL**	**Class-IL**
Baselines	*None – lower bound*	*87.19 (± 0.94)*	*59.21 (± 2.04)*	*19.90 (± 0.02)*
	Offline – upper bound	*99.66 (± 0.02)*	*98.42 (± 0.06)*	*97.94 (± 0.03)*
Task-specific	XdG	99.10 (± 0.08)	-	-
Regularization	EWC	98.64 (± 0.22)	63.95 (± 1.90)	20.01 (± 0.06)
	Online EWC	99.12 (± 0.11)	64.32 (± 1.90)	19.96 (± 0.07)
	SI	99.09 (± 0.15)	65.36 (± 1.57)	19.99 (± 0.06)
Replay	LwF	99.57 (± 0.02)	71.50 (± 1.63)	23.85 (± 0.44)
	DGR	99.50 (± 0.03)	95.72 (± 0.25)	90.79 (± 0.41)
	DGR+distill	99.61 (± 0.02)	96.83 (± 0.20)	91.79 (± 0.32)
Replay + Exemplars	iCaRL (budget = 2000)	-	-	94.57 (± 0.11)

Table 2: Idem sur le protocole *permuted MNIST*

Table 5: Idem as Table 4, except on the permuted MNIST task protocol.

Approach	**Method**	**Task-IL**	**Domain-IL**	**Class-IL**
Baselines	*None – lower bound*	*81.79 (± 0.48)*	*78.51 (± 0.24)*	*17.26 (± 0.19)*
	Offline – upper bound	97.68 (± 0.01)	97.59 (± 0.01)	97.59 (± 0.02)
Task-specific	XdG	91.40 (± 0.23)	-	-
Regularization	EWC	94.74 (± 0.05)	94.31 (± 0.11)	25.04 (± 0.50)
	Online EWC	95.96 (± 0.06)	94.42 (± 0.13)	33.88 (± 0.49)
	SI	94.75 (± 0.14)	95.33 (± 0.11)	29.31 (± 0.62)
Replay	LwF	69.84 (± 0.46)	72.64 (± 0.52)	22.64 (± 0.23)
	DGR	92.52 (± 0.08)	95.09 (± 0.04)	92.19 (± 0.09)
	DGR+distill	97.51 (± 0.01)	97.35 (± 0.02)	96.38 (± 0.03)
Replay + Exemplars	iCaRL (budget = 2000)	-	-	94.85 (± 0.03)

Avantages/Inconvénients de notre approche

Avantages:

- Permet une inférence de tâche et l'utilisation d'un robuste TcH pour du CL quand les ids des tâches ne sont pas connues.
- Infère les embeddings de tâches: Taille fixe, pas besoin d'étendre le réseau avec le nombre de tâches
- Utiliser le DGR pour éviter l'oubli catastrophique sur l'inférence d'embedding de tâches
- Embeddings de tâches: espace continue permet plus de flexibilité pour représenter les tâches et permet un changement graduel entre les tâches (ex. changement dans la distribution des données)

Inconvénient:

- Difficile d'apprendre à approximer des embeddings statiques ancrés pour une tâche t

Conclusion

- Les meilleurs modèles de CL connues à ce jour sont les modèles **DGR** et **Hypernetworks** qui comme nous avons vu sont très bon en Task-IL, Domain-IL et en Class-IL car ils respectent très bien les contraintes mémoire et évitent au maximum le catastrophique forgetting

- Notre contribution de combiner ces 2 approches et d'y ajouter un régresseur d'embeddings de tâches pourrait être une solution encore meilleure au problème du CL.

Références

[1] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. 1989.

[2] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks, 2017.

[3] J. Schwarz, J. Luketina, W. M Czarnecki, A. Grabska-Barwinska, Y. Whye Teh, R. Pascanu, and R. Hadsell. Progress & compress: A scalable framework for continual learning, 2018.

[4] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence, 2017.

[5] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2015

[6] Z. Li and D. Hoiem. Learning without forgetting, 2017.

[8] S-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H Lampert. iCaRL: Incremental classifier and representation learning, 2017.

[21] H. Shin, J. K Lee, J. Kim, and J. Kim. Continual learning with deep generative replay, 2017.

[12] J. von Oswald, C. Henning, J. Sacramento, B. F. Grewe. Continual learning with hypernetworks, oct 2019.

[14] G. M. van de Ven and A. S. Tolias. Three scenarios for continual learning, 2019.

[15] Jaehong Yoon, Eunho Yang, Jeongtae Lee, Sung Ju Hwang. Lifelong Learning with Dynamically Expandable Networks, 2018.

[16] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, Raia Hadsell. Progressive Neural Networks, 2016.

[11] D. Ha, A. M. Dai, and Q. V. Le. Hypernetworks, 2017.

[12] J. von Oswald, C. Henning, J. Sacramento, B. F. Grewe Continual learning with hypernetworks, october 2019.